Este es un manual que nos muestra de una forma rápida de lo que Python puede hacer, encontraremos una pequeña resumen de lo que uno puede hacer. Es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Utilizaremos funciones básicas y esenciales para crear el juego TOQUE O FAMA como la utilización de variables, DEF para crear nuevas instrucciones, WHILE para repeticiones, CADENAS DE CARACTERES, etc.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).

. El intérprete de Python estándar incluye un modo interactivo, en el cual se escriben las instrucciones en una especie de shell: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como también para los programadores más avanzados: se pueden probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.

Existen otros programas, tales como IDLE,[19] bpython[20] o IPython,[21] que añaden funcionalidades extra al modo interactivo, como el auto-completar código y el coloreado de la sintaxis del lenguaje.

Ejemplo del modo interactivo:

```
>>> 1+1

2

>>> a = range(10)

>>> print a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Variables

Las variables se definen de forma dinámica. El signo igual (=) se usa para asignar valores a las variables:

x = 1

x = 'texto' # esto es posible porque los tipos son asignados dinámicamente

Tipos de datos

Los tipos de datos se pueden resumir en esta tabla:

Tipo	Clase	Notas	Ejemplo
str	String	Inmutable	'Integrador'
unicode	String	Versión Unicode de str (eliminado en Python3[25])	u'Integrador'
list	Secuencia	Mutable, puede contener diversos tipos	[4.0, 'string', True]
tuple	Secuencia	Inmutable	(4.0, 'string', True)
set	Conjunto	Mutable, sin orden, no contiene duplicados	set([4.0, 'string', True])
frozenset	Conjunto	Inmutable, sin orden, no contiene duplicados	frozenset([4.0, 'string', True])
dict	Mapping	Grupo de pares clave, valor	{'key1': 1.0, 'key2': False}
int	Número entero	Precisión fija, convertido en long en caso de overflow.	42
long	Número entero	Precisión arbitraria	42L ó 456966786151987643L
float	Número decimal representado en	coma flotante de doble precisión	3.1415927
bool	Booleano	Valor booleano verdadero o falso	True o False

- Mutable: si su contenido (o dicho valor) puede cambiarse en tiempo de ejecución.
- Inmutable: si su contenido no puede cambiarse en tiempo de ejecución.

Tuplas y Listas

- Para declarar una lista, basta usar los corchetes [ ], mientras que para declarar una tupla es recomendable usar los paréntesis (). En ambas los elementos se separan por comas, y en el caso de las tuplas es necesario que tengan como mínimo una coma.
- Tanto las listas como las tuplas pueden contener elementos de diferentes tipos. No obstante las listas suelen usarse para elementos del mismo tipo en cantidad variable mientras que las tuplas se reservan para elementos distintos en cantidad fija.
- Para acceder a los elementos de una lista o tupla, se utiliza un índice entero. Se pueden utilizar índices negativos para acceder elementos a partir del final.
- Las listas se caracterizan por ser mutables, es decir, se puede cambiar su contenido, mientras que no es posible modificar el contenido de una tupla ya creada, puesto que es inmutable.

```
>>> lista = ['aaa', 1, 90]

>>> lista[-1]

90

>>> lista[0] = 'xyz'

>>> lista[0:2]

['xyz', 1]



>>> tupla = (1, 2, 3)

>>> tupla[0] = 2

 ( genera un error )

>>> tupla[0]

1

>>> otratupla = (tupla, ('a', 'b')) # es posible anidar tuplas

>>> 1, 2, 3, 'a' # Esto también es una tupla, aunque es recomendable ponerla entre paréntesis
```

>>> (1) # Aunque entre paréntesis, esto en cambio no es una tupla, ya que no posee una coma tras el elemento

>>> (1,) # En cambio, en este otro caso, sí.

>>> (1, 2) # Con más de un elemento no es necesaria la coma final...

>>> (1, 2,) # ...aunque agregarla no modifica el resultado. Esta tupla

Módulos

Existen muchas propiedades que se pueden agregar al lenguaje importando módulos, que son minicódigos (la mayoría escritos también en Python) que llaman a los recursos del sistema. Un ejemplo es el módulo Tkinter, que permite crear interfaces gráficas que incluyan botones, cajas de texto, y muchas cosas que vemos habitualmente en el sistema operativo. Otro ejemplo es el módulo que permite manejar el sistema operativo desde el lenguaje. Los módulos se agregan a los códigos escribiendo import seguida del nombre del módulo que queramos usar. En este código se muestra como apagar el ordenador desde Windows.

```
apagar = "shutdown /s"
import os
os.system(apagar)
```